آموزش زبان برنامه نویسی جاوا

کنترل سطح دسترسی در جاوا

جلسه شانزدهم

نویسنده : رحمان زارعی

جاوا را ساده،آسان و شیرین بنوشید!!!!

گاهی در زبان جاوا نیاز هست راه دسرسی به ویژگی ها و رفتار های (متغیرها و متدها) کلاس خود را کنترل کنیم.یعنی اجازه ندهیم هر کسی از راه رسید از این متدها و متغیر های کلاسمون استفاده کنه،حالشو ببره بدون این که از ما اجازه بگیره!!!!! کنترل سطح دسرسی به ویژگی ها و رفتار های کلاس را می توان به یک تلفن تشبیه کرد!!! اگر ما تلفن خود را در معرض عموم مردم قرار بدهیم ،همه از آن استفاده می کنند!!! چیزی شبیه تلفن عمومی می شود! اما اگر تمایل نداشته باشیم که هر کسی از آن استفاده کند آن را در مکانی می گذاریم که دسرسی ان برای عموم مردم محدود باشد!!! مثال دومی که می توان برای کنترل سطح دسرسی زد همان چت کردن در شبکه های اجتماعی می باشد!! مثلا شما در واتس اپ می توانید در یک گروه عمومی با سایر دوستان خود چت کنید و همه اعضای گروه از پیام شما با خبر می شوند اما اگر خواستید بصورت کنترل شده و خصوصی چت کنید به pv یا private هم دیگر می روید و هرکسی هم نمی تواند به pv شما دسرسی داشته باشد! همه این مثال ها را گفتم که کنترل سطح دسرسی را برای شما ملموس تر کرده باشم.حالا برویم سراغ اصل مطلب ☺

در جاوا برای کنترل سطح دسرسی از واژه ای تحت عنوان **اصلاح کننده های** سطح دسرسی (Access level **modifiers**) استفاده میکنیم.خب این یعنی چه؟! مثلا برای کنترل سطح دسرسی به تلفن خود و محدود کردن دسرسی به ان،تلفن را از یک مکان عمومی به یک مکان خصوصی می بریم که در دسرس مردم نباشد یعنی دسرسی به تلفن خود را از حالت عمومی به خصوصی تغییر یا اصلاح کردیم به این حالت عمومی (public) یا خصوصی (private) **اصلاح کننده** (**modifiers**) سطح دسرسی می گویند.

اصلاح کننده های (تغییر دهنده ها یا **modifiers**) سطح دسرسی در جاوا تعیین می کنند که سایر کلاس ها تا چه اندازه می توانند از ویژگی ها و رفتار های کلاس فعلی شما استفاده کنند.

پس تا اینجا این اصلاح کننده ها برای کنترل دسرسی سایر کلاس ها به ویژگی ها و رفتار های شما می باشند.در ادامه آموزش به هر یک از اصلاح کننده های سطح دسرسی می پردازیم:

اصلاح کننده سطح دسرسی عمومی – کلمه کلیدی public

- کلمه کلیدی **public** از شناخته شده ترین کلمه های کلیدی جاوا هست و زیاد می بینیمش و باهاش کار کردیم و کار داریم!!!
- public از ساده ترین اصلاح کننده های سطح دسرسی در جاوا می باشد زیرا همان طور که از اسمش معلومه ماهیتی مشخص دارد ،دسرسی ان چیزی شبیه تلفن عمومی است یعنی همه کلاس های دیگر می توانند به متغیرها و متدهایی که از نوع public تعیین کردیم دسرسی داشته باشند.
- پس وقتی یک متغیر یا متد از نوع public در کلاس خود تعریف میکنیم برای سایر کلاس ها این ویژگی ها و متدها قابل دسرسی می باشد.
- در کل یک کلاس ، متغیر ، متد ، اینترفیس (درآینده بررسیش میکنیم) ، سازنده و... وقتی public اعلام می شود،توسط سایر کلاس ها قابل دسرس هستند.
- این کلمه کلیدی زمانی مفید است که متغیر ها و متدهای یک کلاس در یک اپلیکیشن باید قابل دسرس برای کل کلاس های برنامه ما باشد.
- طبق روال معمول وقتی نیاز هست متغیرهای یک کلاس در همه جای اپلیکیشن به اشتراک و استفاده سایر کلاس ها قرار گیرد،آنان را عمومی (public) اعلام می کنیم.
- خب امیدوارم توضیحاتم جامع و کامل و کافی بوده باشه،حالا این اصلاح کننده را با مثال بررسی میکنیم:

مثال: در زیر دو کلاس به نام های **MathExample** و **MainTest** در پکیج **javalike** داریم:

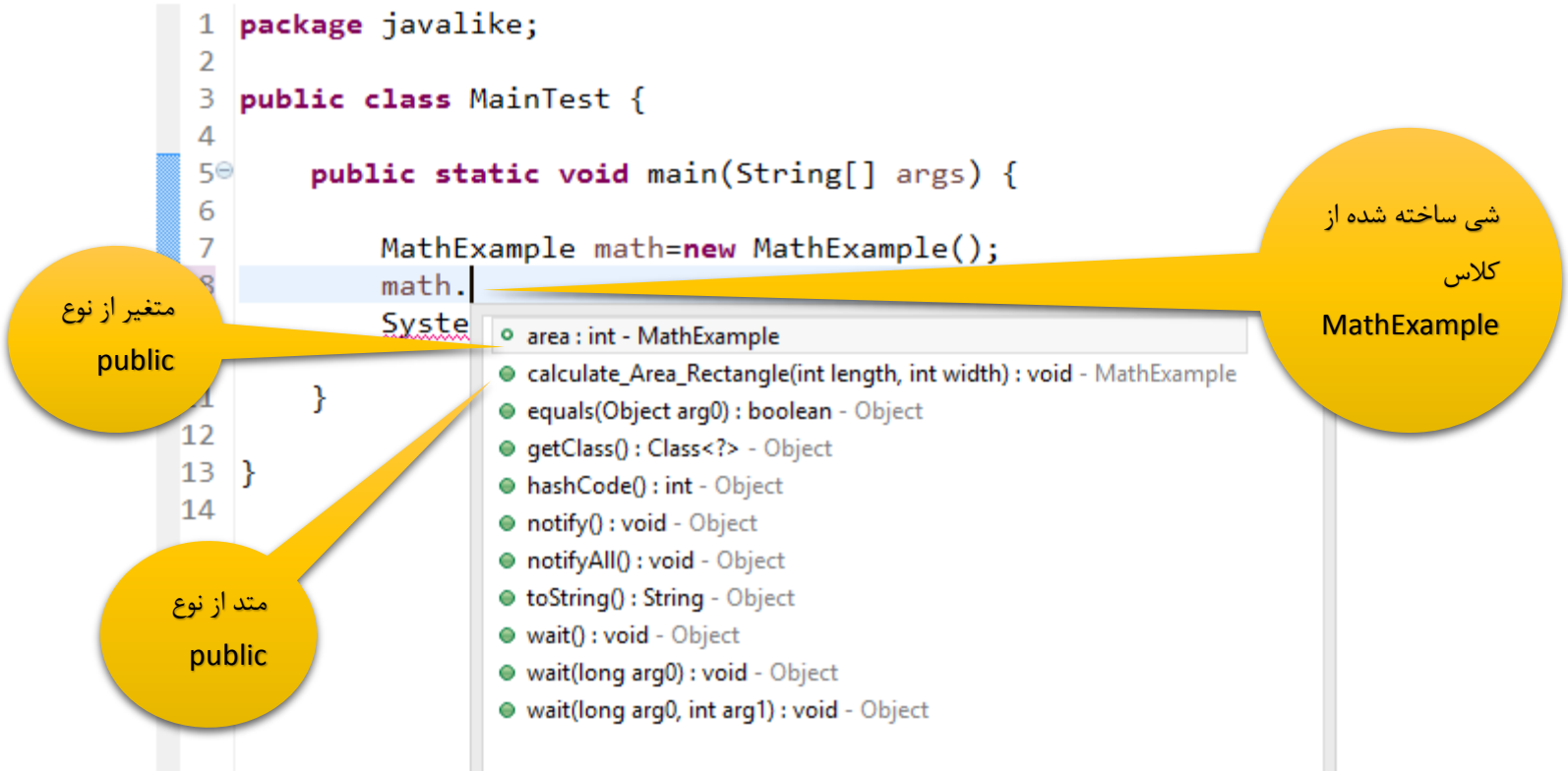
```
package javalike;

public class MathExample {

	public int area;

	public void
calculate_Area_Rectangle(
int length, int width) {

		area = length * width;


	}
```

```
package javalike;

public class MainTest {

	public static void main(String[] args) {

		MathExample math=new MathExample();
		math.calculate_Area_Rectangle(10, 100);
		System.out.println("Area="+math.area);

	}

}
```

- کلاس **MathExample** یک ویژگی به نام area از نوع public و یک متد که مساحت مستطیل را حساب میکند و از نوع public تعریف شده است، دارد. ما قصد داریم در کلاس MainTest یک شی از کلاس MathExample بسازیم و از طریق ان به متدها و متغیر های کلاس MathExample دسرسی داشته باشیم.این یعنی قصد داریم در

یک کلاس از ویژگیها و رفتار های کلاس دیگری استفاده کنیم.از ان جهت که متغیرها و متدهای کلاس MathExample از نوع public هست پس سطح دسرسی عمومی است و به راحتی می توان به ویژگی ها و متد آن در کلاس دیگر نظیر MainTest استفاده کنیم.

- در کلاس MainTest یک شی از کلاس MathExample به نام math ایجاد کرده ایم.
- حال از طریق شی math به ویژگی ها و رفتار های کلاس MathExample درسی پیدا میکنیم.البته درصورتی می توانیم در یک کلاس از ویژگی ها و رفتارهای کلاسی دیگر استفاده کنیم که این ویژگی ها و رفتار ها از نوع public باشند.
- در تصویر(۱) سطح دسرسی شی math که از نوع کلاس MathExample می باشد ، در کلاس MainTest مشاهده می کنید.به راحتی این شی می تواند بدون هیچ مانعی به ویژگی ها و رفتار های کلاس MathExample که از نوع public می باشد دسرسی داشته باشد:



تصویر(۱)

- همان طور که از متد `calculate_Area_Rectangle` مشخص هست که متد دو پرامتر از نوع int از ورودی میگیرد و مساحت مستطیل را محاسبه و درون متغیر نمونهarea میریزد و در پایان در خروجی این متغیر نمونه را چاپ میکنیم.
- حالت قرار گیری دو کلاس در یک پکیج و خروجی آن را در زیر مشاهده می کنید:تصویر(۲)

- شرط لازم استفاده یک کلاس از ویژگی ها و رفتارهای کلاس دیگر هم پکیج بودن دو کلاس می باشد،یعنی هر دو کلاس در یک پکیج باشد،مثلا دو کلاس بالا هر دو در یک پکیج به نام javalike بودند،حالا شرط بعدی (کافی) بستگی به کنترل سطح دسرسی که برای متغیرها و متدهای خود مشخص کردیم دارد.
- اگر دو کلاس هم پکیج نبودند و قصد داشتیم از ویژگی ها و رفتار های کلاس دیگر استفاده کنیم کافیست کلاس مورد نظر درون پکیج مربوطه را import کنیم:

مثال:فرض کنید کلاس A درون پکیج javalike و کلاس B درون پکیج tutorial باشد،قصد داریم در کلاس B یک شی از کلاس A ایجاد کنیم،اما چون کلاس A و B درون پکیج های متفاوتی هستند نمی توان از کلاس A درون کلاس B شی ساخت!تنها راهش این هست که پکیج کلاسA یعنی javalike را درون کلاس B صدا بزنیم(import)کنیم:

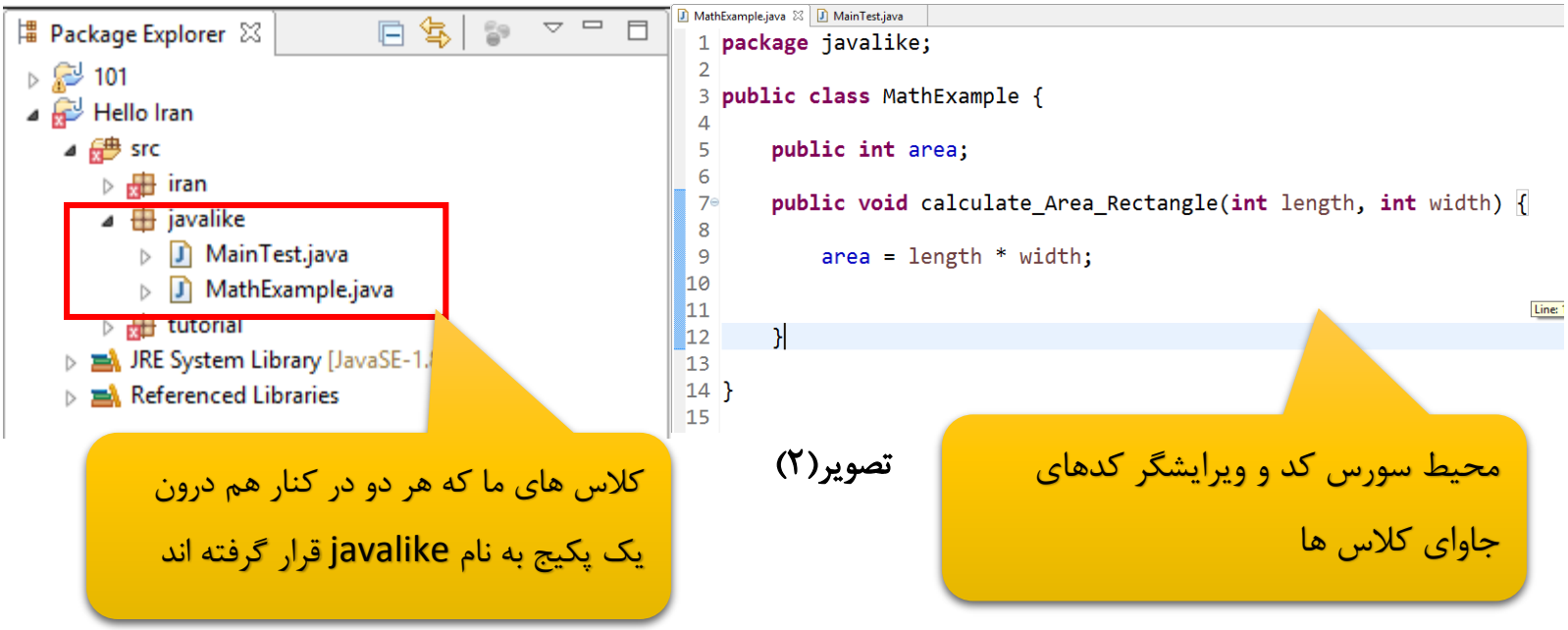
class A

```
Package javalike;


public class A {
	public int a=5;

}
```

class B

```
package tutorial;

import javalike.A;

public class B {

	public static void main(String[] args) {
		A objectA=new  A();

	}
}
```

ادامه اموزش را در پایین دنبال می کنیم.....



تصویر(۲)

خروجی مثال بالا:خروجی مثال دو کلاس MathExample وMainTest بصورت زیر است

```
Area=1000
```

❖ با توجه به این که متد main در کلاس MainTest بود و یک شی از کلاس MathExample درون کلاس MainTest ایجاد کرده بودیم کافیست کلاس MainTest را اجرا کنیم و بصورت خودکار هر دو کلاس اجرا می شود.

- اصلاح کننده سطح دسرسی public کمترین حافظت را از ویژگی ها و رفتار های ما میکند.تنها زمانی از این اصلاح کننده استفاده کنید که کاملا اطمینان داشته باشید که نیاز هست هرچیز و همه چیز به متغیر ها و متدهای ما دسرسی داشته باشند.
- در این مثال با روش دسرسی به ویژگی ها و رفتارهایی که از نوع public هستند در کلاس دیگر یاد گرفتیم.حالا سراغ اصلاح کننده**(modifiers)** دیگر در جاوا می رویم.

اصلاح کننده سطح دسرسی پیشفرض(Default) – بدون کلمه کلیدی

اصلاح کننده **(modifiers)** سطح دسرسی پیشفرض (Default) به معنای این هست که ما صریحا برای کلاس ها ، متدها ، متغیرها و.... اصلاح کننده دسرسی در نظر نمی گیرید یا درکل هیچ کلمه کلیدی برای کلاس ها،متغیرها و...تعریف نمی کنیم.

وقتی متغیرها یا متدهای یک کلاس اصلاح کننده پیشفرض درنظر گرفته می شود برای سایر کلاس ها قابل دسرس هستند.

```
package javalike;

public class A {
	int a=5;
	int b=123;
	String str=" Salam";

}
```

- Modifier یا اصلاح کننده سطح دسرسی برای متغیرهای این کلاس از نوع پیشفرض هستند چون هیچ کلمه کلیدی برای متغیرها تعریف نشده است.

اصلاح کننده (Modifier) سطح دسرسی خصوصی – کلمه کلیدی private

- وقتی متدها ، متغیرها ، سازنده ها و... با کلمه کلیدی private اعلام می شوند،تنها درون بدنه کلاس خودشان قابل دسرس هستند.
- با private کردن ویژگیها و رفتار های کلاس خود آن را از دسرس هر نامحرمی بدور نگه میدارید ☺
- Modifier یا اصلاح کننده (تغییر دهنده) سطح دسرسی private ، سطح دسرسی را بسیار محدود می کند.

- کلاس ها (class) و اینترفیس ها (interface) هرگز نمی توانند از نوع private باشند.
- در مورد اینترفیس در جلسات آینده که مربوط به مباحث شی گرایی می باشد خواهیم پرداخت.
- متغیرهایی که private تعریف می شوند تعریف می شوند تنها داخل بدنه کلاس قابل دسرس هستند و در کلاس های دیگر بصورت مستقیم قابل دسرس نیستند.
- برای دسرسی یک کلاس به متغیر های private درون کلاس دیگر از متدهای public، getter استفاده می کنیم.
- متدهای getter چی هستند؟!!!! جلوتر بهش می پردازیم! ☺

کاربرد اصلاح کننده private

کلمه کلیدی private شی ها و داده های درون یک کلاس را از محیط بیرون حفاظت می کند.

مثال:

```
package javalike;

public class A {
	private int a=5;
	private String str=" Salam";
}
```

- متغیرهای این کلاس از نوع اصلاح کننده private تعریف شده اند.

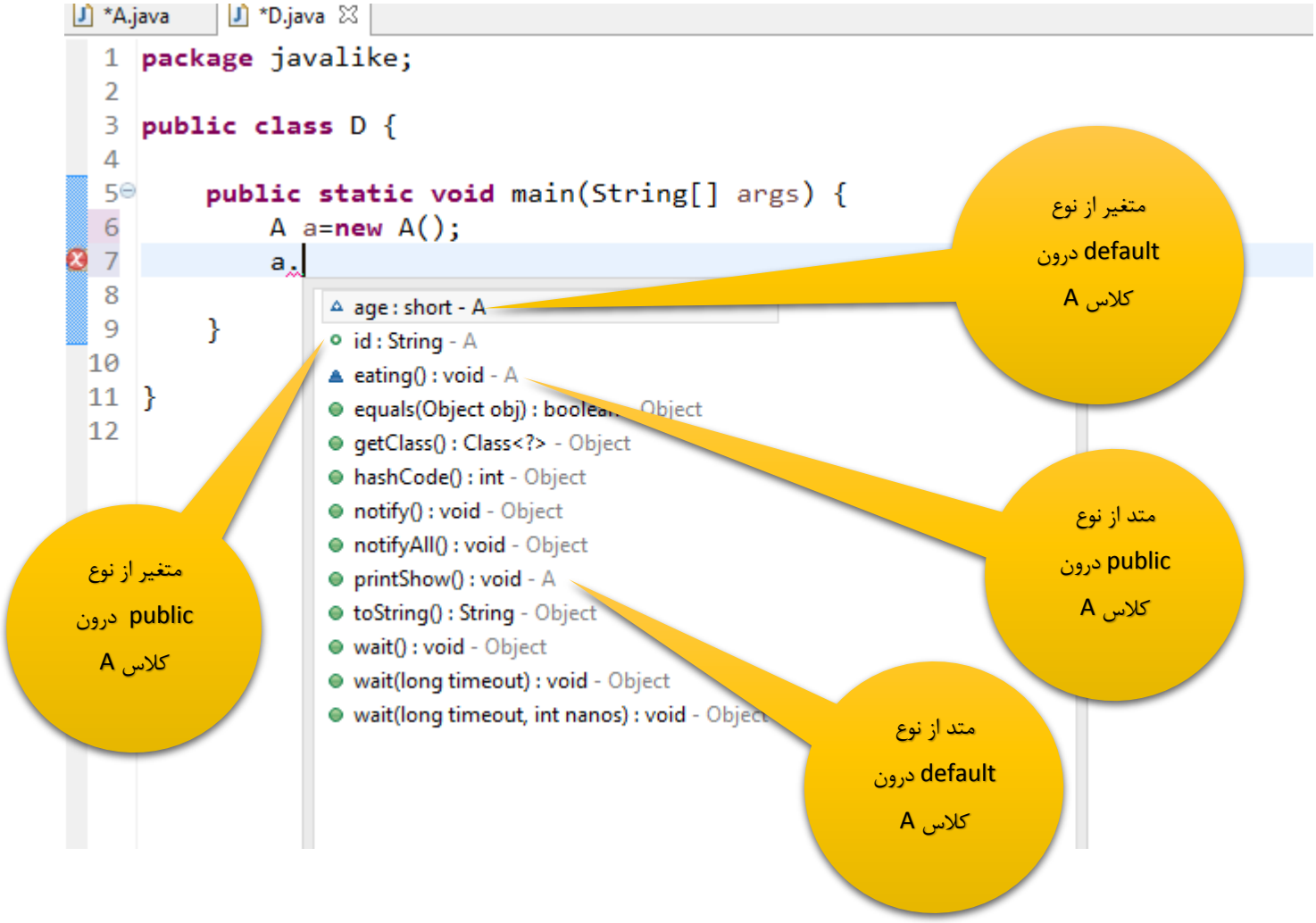
مثال:در مثال زیر دو کلاس A,D که هم پکیج هستند و هر دو درون پکیج javalike قرار دارند. برخی متغیرها و متدهای کلاس A از نوع private و برخی دیگر از نوع public و برخی از نوع پیشفرض تعریف شده اند.حال قصد داریم در کلاس D با ساختن شی از کلاس A کنترل سطح دسرسی ها را بررسی کنیم:

کلاس A :

*A.java *D.java

```
1  package javalike;
2
3  public class A {
4      private int length = 5;
5      private String str = " Salam";
6      public String id = "921115614";
7      short age = 22;
8
9      public void printShow() {
10
11         System.out.println("");
12
13     }
14
15     private void sleeping() {
16
17         System.out.println("");
18
19     }
20
21     void eating() {
22
23         System.out.println("");
24
25     }
26
27 }
28
```

کلاس D :در کلاس D یک شی از کلاس A به نام a ایجاد کرده ایم.حالا از طریق شی a قصد داریم به متغیرها و متدهای کلاس A دسرسی داشته باشیم،فراموش نکنید که ما در خارج از کلاس A و درون کلاس D قصد داریم به متغیرها و متدهای کلاس A دسرسی پیدا میکنیم.همان طور که در تصویر(۳) مشاهده میکنیداین شی تنها به ویژگی ها و متدهای public و default دسرسی دارد و نمی تواند به متدها و متغیرهای private دسرسی پیدا کند.



تصویر(۳)

- همان که در تصویر(۳) مشخص هست متدها و متغیرهای از نوع private بصورت مستقیم قابل دسرس نبودند.
- من در دستور چاپ در مثال بالا علامت شکلک گذاشتم جدی نگیریدش!!!! ،شما اگر میخواید این کد رو تست کنید بصورت زیر عمل کنید:دو کلاس A,D را در یک پکیج کنار هم قرار بدهید.

```
package javalike;

public class A {
	private int length = 5;
	private String str = " Salam";
	public String id = "921115614";
	short age = 22;

	public void printShow() {

		System.out.println(":----------)");

	}

	private void sleeping() {

		System.out.println(";-)");

	}

	void eating() {

		System.out.println(":)");

	}

}
```

```
package javalike;

public class D {

	public static void main(String[] args) {
		A a=new A();
		a.eating();
		a.printShow();
		System.out.println(a.id);
		System.out.println(a.age);

	}

}
```

خروجی:با Run کردن کلاس D خروجی بصورت زیر است:

```
:)
:----------)
921115614
22
```

اصلاح کننده (Modifier) سطح دسرسی protected – کلمه کلیدی protected

این سطح دسرسی و کلمه کلیدی protected مربوط به مباحث ارث بری در جاوا می شود که در جلسات اینده مفصل به این مبحث می پردازیم.

پیروز و موفق باشید
